LA-UR-23-23992

Approved for public release; distribution is unlimited.

Title:	An evaluation of risks associated with relying on Fortran for mission critical codes for the next 15 years
Author(s):	Shipman, Galen M. Randles, Timothy C.
Intended for:	Report
Issued:	2023-04-18 (rev.1)

An evaluation of risks associated with relying on Fortran for mission critical codes for the next 15 years

LA-UR-23-23992
April 18, 2023

Los Alamos
NATIONAL LABORATORY

EXECUTIVE SUMMARY

This document examines risks associated with relying on Fortran for codes critical to nuclear security into the late 2030s. The study was requested by the LANL ASC program office to support its decisions. To date, much of the discussion about the prospects for Fortran has tended to be informal and centered on questions (such as, "will the language be around in 20 years") that are impossible to answer with any certainty. By focusing on estimates of the likelihood associated with different outcomes, rather than on specific predictions, we hope to clarify points of disagreement within the community and strengthen the analytic basis for reasoning about the future.

Our assessment for seven distinct risks associated with continued use of Fortran are that in the next fifteen years:

1. It is *very likely* that we will be unable to staff Fortran projects with top-rate computer scientists and computer engineers.

2. There is an *even chance* that we will be unable to staff Fortran projects with top-rate computational scientists and physicists.

3. There is an *even chance* continued maintenance of Fortran codes will lead to expensive human or financial maintenance costs.

4. It is *very unlikely* that codes that rely on Fortran will have poor performance on future CPU technologies.

5. It is *likely* that codes that rely on Fortran will have poor performance for GPU technologies.

6. It is *very likely* that Fortran will preclude effective use of important advances in computing technology.

7. There is an *even chance* that Fortran will inhibit introduction of new features or physics that can be introduced with other languages.

Here we have followed the convention adopted by the Intelligence Community (ICD-203) for probabilistic language.

almost no chance	*very unlikely*	*unlikely*	*even chance*	*likely*	*very likely*	*almost certain*
01-05%	05-20%	20-45%	45-55%	55-80%	80-85%	95-99%

A more complete description of the meaning of our judgments, along with supporting reasoning, is given in the main document.

Our assessments lead us to the view that continued use of Fortran in our mission critical codes poses unique challenges for LANL. While Fortran will continue to be supported at some level, particularly on CPU-based systems, the outlook for advanced technology systems is dim. The ability to leverage broader and more modern open-source technologies / frameworks is unlikely, increasing the cost of new physics and feature development.

The vendor ecosystem of Fortran compilers is worrying. Intel and GCC communities have the most robust Fortran compilers for modern Fortran (Fortran 2008) on CPU technologies but have less mature support for GPU technologies. Nvidia has good support for GPU technologies but lacks support for modern Fortran needed by LANL. Open-source efforts around an LLVM compiler for Fortran, known as Flang, are inadequate to meet either requirement (robust support for modern Fortran and GPU technologies). Complicating things further, there are competing Fortran technologies for GPUs including standards such as OpenACC and OpenMP and vendor proprietary technologies such as Cuda Fortran. While similar diversity exists for other languages (such as C++) there are no infrastructures for portability like Raja and Kokkos for Fortran.

From the perspective of continued technological advancement, Fortran receives much less attention from industry and academia relative to other languages. C++ and Python have numerous examples of advances to support massive on-node parallelism at the language and library levels. Some of these advancements have recently found their way into language standards with support from multiple proprietary and open-source compilers. Experience over the past two decades has shown a much slower evolution of the Fortran language and significant delays in supporting new language standards in compilers. From the perspective of market demand, Fortran can be considered a niche or legacy technology. This is evident by multiple metrics including language popularity analysis and number of job postings on major employment websites.

Ability to Staff

We judge it is *very likely* that we will be unable to staff Fortran projects with top-rate computer scientists and computer engineers, and that there is an *even chance* we will be unable to staff Fortran projects with top-rate computational scientists and physicists.

- "Fortran is no longer widely taught to university students or valued as a useful skill by industry. Consequently, adoption of new users has been stagnating, large scientific Fortran projects have been migrating to other languages, and the communities of Fortran programmers remained scattered and isolated." [StateOfFortran2022]. Universities are increasingly using higher level languages, such as Python, to teach computational science and engineering. Computer science and computer engineering graduates are rarely if ever introduced to Fortran during their coursework and are more likely to be introduced to C, C++, Python, and functional programming such as Scheme.
- The broader job market has a significantly lower demand for Fortran developers. As of this writing, 1,299 U.S.-based jobs posted to Indeed.com listed Fortran in their job description compared to 47,919 listing C++ and 128,745 listing Python. From the perspective of hiring, it has been difficult to hire and staff large-scale Fortran code bases with individuals with computer science or computer engineering backgrounds. Computational science and physics students are sometimes introduced to Fortran, largely using legacy codes to conduct their research.
- Many of the larger scale physics codes have migrated or been replaced by C++ codes and the use of Python as an analysis language now dominates, which may change the composition of skills moving forward.

It should be noted that training staff in the use of Fortran is not a major challenge if the staff member has sufficient experience in another programming language. Attracting (and retaining) staff in these large Fortran projects may prove more difficult. It is also possible that as the pool of Fortran developers continues to decrease, the demand for this skill set on legacy code bases across the industry will remain flat for quite some time, meaning increased competition for the relatively few developers with deep Fortran expertise. This has the potential to further erode retention and our ability to compete on salary.

Contrasting Reviewer Judgments

Two reviewers disagreed with aspects of this judgment, indicating that a variety of other factors have greater impact on our ability to attract top CS / CE talent.

Contrasting Reviewer Judgments

Two reviewers disagreed with aspects of this judgment and wrote:
1) Even though it is easy for staff to learn Fortran if they have sufficient experience in another programming language, we find that they usually don't/won't learn it at a very deep level because they don't see themselves being involved in that area of the code for a long enough period of time to make it worth their time or see that as a marketable enough skill set to learn at that deep level.
2) While learning Fortran is not too difficult, few potential employees will be interested in it. Even if you tell them that you are willing to train them and provide the time they will need to learn Fortran as a cost of doing business, few people are interested in learning an arcane skill with low marketability.

Cost of maintenance

We judge there is an *even chance* continued maintenance of Fortran codes will lead to expensive human or financial maintenance costs.

- Compiler technology costs are significant for Fortran and are trending upwards. Estimates of tens of millions of dollars have been discussed recently. Details on why this is the case are covered in the next section.
- The cost to train staff in Fortran is more than likely a small overall cost; the much more significant cost is the training of staff in our application structure (LAP and EAP in particular) where the complexity of these codes can require many months or even years of training before staff are conversant. This cost is not considered in our assessment because it is not so much a consequence of a particular language as of the complexity of the application code.
- Funding of development of interfacing among languages is significant and time consuming and often requires multiple full-time staff members to make sweeping changes across the code base to support inter-operability.

The last consideration only applies if there are investments in language inter-operability or use of advances in hardware technologies. Costs for just the maintenance of our existing codes is likely comparable with that for other languages.

Contrasting Reviewer Judgments

One reviewer disagreed with this judgment and wrote:

> We face a virtual certainty that continued maintenance of Fortran codes will entail expensive human or financial maintenance costs. Cost of the language infrastructure is part of the maintenance cost and, as you mention, the cost for ensuring working Fortran compilers is high. For C++ (and C and Python and others), industry bears most of that cost and, at most, DOE needs to cover optimizations for some very specific use cases. Further, once they are implemented, they tend to be maintained as part of the language infrastructure for which industry pays.

Ability to make use of advances in hardware

We judge it is *very unlikely* that codes that rely on Fortran will have poor performance on future CPU technologies, it is likely that codes that rely on Fortran will have poor performance for GPUs, and it is *very likely* that Fortran will preclude effective use of important advances in computing technology.

- Fortran has historically provided a very high level of performance on CPU-based architectures. In some cases, Fortran can be optimized more effectively than equivalent C or C++ code. This is particularly true for serial and even data parallel / vectorized code.
- Major technology provider's compilers such as Intel (ifx) and Nvidia (nvfortran) are likely to continue to provide high levels of performance in the future. Open-source compilers such as GCC's gfortran will likely continue to lag somewhat behind these vendor optimized compilers. The LLVM-based Fortran compiler project, Flang, will likely lag even further behind GCC for the foreseeable future.
- In contrast to the case for CPUs, support for GPU technologies tends to significantly lag in support for other languages (C/C++). Robust compiler technologies are necessary to make use of advances in hardware technologies. The Frontier and El Capitan systems are two examples in which the Fortran compiler technology has lagged significantly behind other compiler technologies (C / C++). Multiple competing standards for Fortran-based GPU programming with varied levels of robustness and support exist today (Fortran OpenMP Target offload and OpenACC). Neither of these technologies is robustly supported on the AMD GPU (MI250) today.
- Efforts to fill this gap are ongoing. The Exascale Computing Project (ECP) is funding an open-source Fortran compiler based on LLVM known as Flang, largely in recognition of the lack of robust Fortran compilers for these systems. Other efforts include funding code sorcery (now Siemens) to develop OpenMP target offload and OpenACC backends for the GCC/gfortran compilers. Both efforts are largely reactionary, due to poor community and technology provider support for Fortran on advanced technologies.
- The level of funding required to continue supporting these efforts is not known. Estimates in the tens of millions of dollars have been discussed. Even when a technology provider has an in-house Fortran compiler team that supports advanced hardware, such as the case for the NVIDIA nvfortran compiler, the lack of timely and robust support for "modern Fortran" has proven a major hurdle. As a specific example, efforts to port xRAGE to the PGI Fortran compiler (now rebranded as the nvfortran compiler) have been ongoing since April of 2019 and xRAGE is still unable to be built and run with this compiler. It is important to note that nvfortran still does not support the Fortran 2008 standard, a full 15 years after its ratification by the standards committee.
- The ecosystem of tools, particularly for performance portability, is significantly lacking in Fortran. Several performance portability infrastructures are currently available in the C++ ecosystem including Kokkos (Sandia National Laboratory), RAJA/CHAI (Lawrence Livermore National Laboratory), SYCL (Khronos Group), and OneAPI (Intel). Each of these infrastructures provides the ability to describe iteration patterns over data, computational kernels to execute over the iteration, and mechanisms for managing placement and describing data layout in a potentially heterogenous node architecture. Furthermore, many of the concepts from these infrastructures are influencing the C++ language standard. C++17 has included parallel iterators and C++23 has included n-dimensional arrays (mdspan). While Fortran does have efforts in language level parallelism, such as DO CONCURRENT, implementations make no guarantee of parallelism and support for GPU parallelism is limited to a single vendor specific compiler (nvfortran).

- Beyond GPUs, which may be viewed as commodity technologies as opposed to advanced technologies, Fortran is likely to significantly lag in robust support. Coarse-grained reconfigurable architectures including Data Flow processors and processing near / in memory will undoubtedly support Python and C++ well before they support Fortran. In some cases, it is likely that DOE may be the only proponent of Fortran on these advanced technologies.

The lack of robust support for GPU and other technologies may make certain platforms have very low performance for Fortran codes or effectively lock out our Fortran codes from some vendors. Compilers such as nvfortran will have support for Nvidia GPUs, other vendors such as AMD have relatively poor support for their GPUs using Fortran. The trade space may increasingly become portability versus performance.

Contrasting Reviewer Judgments

One reviewer disagreed with aspects of this judgment and wrote:
Your comments about performance of Fortran codes on CPUs or GPUs are overly broad. First, the performance on either processor type will likely depend on the investment discussed in my first point. If it is high enough, then neither processor type needs to have poor performance for Fortran applications. More importantly, the performance of Fortran codes will heavily depend on the language features that they use. We can likely support good performance on either processor type for codes that largely restrict themselves to Fortran 95, with use of only carefully considered features of later versions of the standard (e.g., the C interface stuff). Codes that insist on using the latest Fortran standard (aggressively? anything beyond the carefully selected features?) will require that investment to be substantial.

Contrasting Reviewer Judgments

One reviewer disagreed with this judgment and wrote:
This paper seems to project a feeling that we have no control over the future of our computing environment. LLNL specs systems that are good for C++ and ignore Fortran, because they made that switch twenty years ago. The UK's Archer2 system was spec'd to run Fortran codes, because over 75% of their cycles are Fortran. We could choose to invest in computers and software environments that are good for our codes and our mission. $10M put towards getting *[company name]* to improve their compiler is nothing compared to the costs of replacing Fortran in our code base.

Inability to introduce new features or physics

We judge there is an *even chance* that Fortran will inhibit introduction of new features or physics that can be introduced with other languages.

- For features and physics developed directly within an existing code base, the use of Fortran is unlikely to cause challenges. For features and physics that can be realized through integration of libraries, the use of Fortran will introduce some overheads, but these are likely to be relatively small in scope (interfacing between Fortran and C++, for instance).
- In other areas, such as the ability to leverage robust frameworks developed by the community, the use of Fortran will pose a greater challenge. Many of the scientific computing frameworks in use today are written primarily in C++ [MFEM, FleCSI, Parthenon], adopting these frameworks in Fortran code bases would prove extremely challenging and would likely negate any benefits.

Most recent computational science frameworks have been developed in C++, a trend that is likely to continue, which will inhibit the adoption of modern techniques / methods without a custom rewrite or adaptation to existing codes. This is the most likely impact: the inability to effectively leverage broader community development projects.

References

[StateOfFortran2022] Kedward, Laurence J.; et al. “The state of fortran.” *Computing in Science & Engineering* 24.2 (2022): 63-72.

[MFEM] mefm.org.

[ECP-status] https://confluence.exascaleproject.org/pages/viewpage.action?pageId=140935589.

[FleCSI] Bergen, Ben; et al. “FleCSI 2.0: The Flexible Computational Science Infrastructure Project.” European Conference on Parallel Processing. Springer, Cham (2022).

[Parthenon Grete, Philipp; et al. “Parthenon—a performance portable block-structured adaptive mesh refinement framework.” arXiv preprint arXiv:2202.12309 (2022).